Splitting the Scheduling Headache

Kevin Foltz Jehoshua Bruck

California Institute of Technology
Mail Code: 136-93
Pasadena, CA 91125

Email: {kfoltz,bruck}@paradise.caltech.edu

Abstract

The broadcast disk provides an effective way to transmit information from a server to many clients. Information is broadcast cyclically and clients pick the information they need out of the broadcast. An example of such a system is a wireless web service where web servers broadcast to browsing clients. Work has been done to schedule the information broadcast so as to minimize the expected waiting time of the clients. This work has treated the information as indivisible blocks that are transmitted in their entirety. We propose a new way to schedule the broadcast of information, which involves splitting items into smaller sub-items, which need not be broadcast immediately after each other. This relaxes the previous restrictions, and hence allows us to have better schedules with lower expected waiting times. We look at the case of two items of the same length, each split into two halves, and show that we can achieve optimal performance by choosing the appropriate schedule from a small set of schedules. We derive a set of optimal schedules and show which one to use, as a function of the demand probabilities. In fact we prove the surprising result that there are only two possible types of optimal cyclic schedules for items 1 and 2. The first starts with 1122 and the second with 122122. For example, with demand probabilities $p_1 = .19$ and $p_2 = .81$, the best order to use in broadcasting the halves of items 1 and 2 is a cyclic schedule with cycle 122122.

1 Introduction

As mobile computing gains popularity, it becomes increasingly important to find efficient methods of communicating with mobile clients. In general, the mobile clients have considerably less outgoing bandwidth than incoming bandwidth, making communication highly asymmetric.

Web browsing is a good example of this situation. A person browsing the web typically receives a lot of information and sends very little. We will describe an efficient way of sending information to portable web browsers. The measure of efficiency that we will use is "expected waiting time". This is the expectation of the idle time between when a client starts waiting for an item and when it has completely received the item, averaged over all items, with relative weight p_i, the demand probability of item i, given to item i in the average. By idle time, we mean time during which the client is not receiving useful data. Thus, "idle time" + "time to send desired data" = "total listening time".

The broadcast disk is a way to send information to many clients at the same time. Using this scheme, data is broadcast through the air in a cyclic fashion. The data is chosen based on how likely it is that clients are going to want it. When a client wants some data, it listens to this broadcast stream until it receives the desired data. If the desired data is not in the broadcast cycle, other means are used to retrieve the data. Essentially, the broadcast disk acts as a common cache for many clients, where data in this cache is made available cyclically, according to the broadcast schedule. The goal is to schedule the broadcast information in a way that minimizes the expected waiting time of the clients.

Vaidya and Hameed [5, 6, 9] worked out the optimal broadcast frequencies of items within a schedule as a function of their demand probabilities, p_i, and lengths, l_i. They showed that to minimize expected waiting time, the frequencies of broadcast, f_i, should be proportional to $\sqrt{\frac{p_i}{l_i}}$. This led to an algorithm that attempted to achieve these relative frequencies. This algorithm is good because it is computationally fairly simple and works for an arbitrary number of broadcast items with arbitrary lengths and demand probabilities.

Jiang and Vaidya [7] discuss ways to minimize the variance of the response time. They also propose a way to trade-off between minimizing the mean and minimizing the variance of the response time. Aksoy and Franklin [1] discuss scheduling the broadcast of information based on client requests. They consider such metrics as average and worst case performance, scheduling overhead, and robustness in the presence of environmental changes. Bestavros [3] describes a way to add fault tolerance to broadcast disks by sending parity information in addition to data. Bar-Noy, Bhatia, Naor, and Schieber [2] look at scheduling in general, and show that there is an optimal cyclic schedule for a broadcast disk, and finding it is NP-hard. Leong and Si [8] discuss how to choose which items to broadcast, using ideas of cache management. Franklin, Zdonik, Acharya, and Alonso [4, 10] also discuss aspects of broadcast disks.

We examine the scheduling of items for a broadcast disk. We represent a broadcast schedule for two items by a sequence of 1's and 2's, where '1' represents the broadcast of the first item, and '2' represents the broadcast of the second item. In this paper, we think of each item as consisting of two halves, so a 1 or 2 will represent one of these halves, and not the entire item. For example, a schedule in which the two items are broadcast in their entirety alternately is 112211221122... and not 121212..., since we need two halves of each item to broadcast the entire item.

Broadcast schedules are cyclic, so we will represent them by one of their cycles. Since we have two halves of each item, we assume these halves are broadcast alternately. The schedule 122, for example, should really be written as $1_A2_A2_B1_B2_A2_B$, where 1_A and 1_B are the two halves of item 1, and 2_A and 2_B are the two halves of item 2. This would more accurately represent one period, but we shorten the representation to 122 with the understanding that the two halves of each item

are broadcast alternately.

Another representation of a schedule uses the number of 2's between consecutive 1's in the schedule. We use a bracketed sequence of numbers that represent the number of 2's between each consecutive pair of 1's. For example, [0,2] represents the schedule 1122, and [0,0,1,3,2] represents 11121222122.

Sometimes we want to indicate that a certain instance of an item may or may not be present in a schedule. Parentheses will be used to indicate the possible presence of an instance of an item in a schedule. For example, a schedule in which we know only that item 2 is never broadcast twice consecutively could be represented as $1(2)1(2)1(2)\ldots$.

We define a useful function dealing with expected waiting times.

Definition 1 *EWT(S,p_1) is the expected waiting time using schedule S with demand probabilities p_1 and $p_2 = 1 - p_1$, assuming two items, each of length one unit, split into two halves.*

This is our metric for evaluating schedules. The lower the value of EWT(S,p_1), the better schedule S is for demand probabilities p_1 and $p_2 = 1 - p_1$.

Vaidya and Hameed assumed a uniform spacing of items within a schedule (which is the optimal spacing, when achievable) to derive their broadcast frequencies. However, in most cases it is not possible to achieve these frequencies with uniform spacing, since the individual schedules for the different items usually do not fit together well. Also, when we broadcast items of varying lengths, we must wait through long items in their entirety even if we just want a short item. This can lead to a large deviation of the expected waiting time from the optimal time.

As an example, consider two items, one that is accessed frequently and another that is accessed less frequently. Suppose $l_1 = 1, p_1 = .2, l_2 = 1, p_2 = .8$. Then, according to Vaidya and Hameed, we want $f_1 = \frac{1}{3}, f_2 = \frac{2}{3}$. We want to broadcast uniformly, so we should schedule items 1 and 2 as in figure 1(a) and (b). If we could do this, we would achieve an expected waiting time of $\frac{9}{10}$.

However, it is impossible to merge these schedules together while preserving the property of uniform spacing. The scheduling algorithm described by Vaidya and Hameed gives us one of the schedules in Figure 1 (c) or (d), depending on how we break ties. These have expected waiting times 1 and $\frac{29}{30}$.

To address the problem of merging schedules, we relax the restriction used by Vaidya and Hameed of broadcasting an item all at once, and look at splitting items into sub-items and scheduling the broadcast of these sub-items. We look at the specific case of two items, each of the same length, each split into two sub-items. We find that the optimal schedule for the example above is the schedule in Figure 1 (e), with expected waiting time $\frac{49}{60}$. Note that this result, when splitting is allowed, is better than even the unachievable theoretical limit of $\frac{9}{10}$ when splitting is not allowed.

In general, we show the following:

Theorem 1 *For two items of the same length, each split into two halves, the broadcast schedule that minimizes expected waiting time is:*

1122, *if* $p_1 \in \left(\frac{5}{16}, \frac{1}{2}\right]$

11222, *if* $p_1 \in \left(\frac{5}{21}, \frac{5}{16}\right]$

112222, *if* $p_1 \in \left(\frac{1}{5}, \frac{5}{21}\right]$

$122122\overbrace{2\ldots2}^{n}, n = Max\left(0, \left\lfloor \frac{-13+\sqrt{-103+\frac{32}{p_1}}}{2} \right\rfloor\right)$, *if* $p_1 \in \left(0, \frac{1}{5}\right]$

The rest of the paper is devoted to this result. We first present some lemmas useful in proving Theorem 1, and then prove the theorem. In the proof, we describe an algorithm which can be

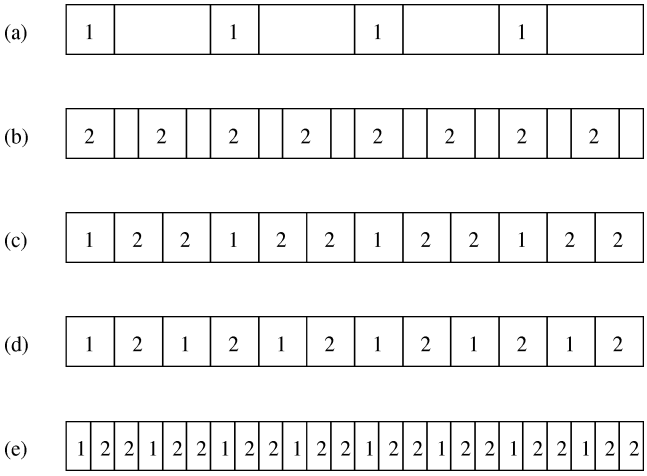

Figure 1: (a) ideal scheduling for item 1, (b) ideal scheduling for item 2, (c) and (d) actual schedules generated using Vaidya and Hameed's algorithm, (e) our schedule using splitting

used to reduce the search for optimal schedules from all schedules to a smaller set of irreducible schedules. We then determine which schedules are in this set. Finally, we compare these schedules with each other and see that the schedules listed in Theorem 1 are optimal on their respective intervals.

2 The Lemmas

These Lemmas provide ways of manipulating schedules into similar but better schedules. In these lemmas, the same applies if we switch item 1 and item 2. Basically, we either name an item "1" or "2" depending on which way makes the schedule in consideration fit the lemmas. Here we present the lemmas. See the Appendix for their proofs.

Lemma 1 *(The Rearrangement Lemma)*
The following hold for all values of p_1*:*
(a)

$$EWT(S', p_1) \leq EWT(S, p_1), \text{ where}$$

$$S = \ldots 1 \overbrace{2 \ldots 2}^{a \geq 0} 1 \overbrace{2 \ldots 2}^{b \geq 3} 1 \overbrace{2 \ldots 2}^{c \geq 0} 1 \ldots 1 \overbrace{2 \ldots 2}^{d \geq 0} 1 \overbrace{2 \ldots 2}^{e \geq 2} 1 \overbrace{2 \ldots 2}^{f \geq 0} 1 \ldots,$$

$$S' = \ldots 1 \overbrace{2 \ldots 2}^{a} 1 \overbrace{2 \ldots 2}^{b-1} 1 \overbrace{2 \ldots 2}^{c} 1 \ldots 1 \overbrace{2 \ldots 2}^{d} 1 \overbrace{2 \ldots 2}^{e+1} 1 \overbrace{2 \ldots 2}^{f} 1 \ldots, \text{ and}$$

$$(a + b + c) - (d + e + f) \geq 1$$

(b)

$$EWT(T', p_1) \leq EWT(T, p_1), \textit{ where}$$

$$T = \ldots 1 \overbrace{22\ldots222}^{n\geq 2} 121 \overbrace{2}^{x=0,1} 1 \overbrace{2}^{y=0,1} 1 \ldots,$$

$$T' = \ldots 1 \overbrace{22\ldots222}^{n} 112 \overbrace{2}^{x} 1 \overbrace{2}^{y} 1 \ldots$$

(c)

$$EWT(U', p_1) \leq EWT(U, p_1), \textit{ where}$$

$$U = \ldots \overbrace{1\ldots1}^{r\geq 1} \overbrace{2\ldots2}^{s\geq 1} \overbrace{11\ldots11}^{t\geq 3} \overbrace{22\ldots22}^{u\geq 3} 1 \ldots,$$

$$U' = \ldots \overbrace{1\ldots1}^{r} \overbrace{2\ldots2}^{s} \overbrace{11\ldots1}^{t-1} 21 \overbrace{2\ldots22}^{u-1} 1 \ldots$$

(d)

$$EWT(V', p_1) = EWT(V, p_1), \textit{ where}$$

$$V = \overbrace{1\ldots1}^{a_1} \overbrace{2\ldots2}^{b_1} \overbrace{1\ldots1}^{a_2} \overbrace{2\ldots2}^{b_2} \ldots \overbrace{1\ldots1}^{a_{k-1}} \overbrace{2\ldots2}^{b_{k-1}} \overbrace{1\ldots1}^{a_k} \overbrace{2\ldots2}^{b_k},$$

$$V' = \overbrace{2\ldots2}^{b_k} \overbrace{1\ldots1}^{a_k} \overbrace{2\ldots2}^{b_{k-1}} \overbrace{1\ldots1}^{a_{k-1}} \ldots \overbrace{2\ldots2}^{b_2} \overbrace{1\ldots1}^{a_2} \overbrace{2\ldots2}^{b_1} \overbrace{1\ldots1}^{a_1}$$

Lemma 2 *(Corollary to Lemma 1 part (a)) If, instead of $S = [a, b, c, \ldots, d, e, f, \ldots]$, S is one of [a, b, c = d, e, f, ...], [a, b = d, c = e, f, ...], [a = f, b, c = d, e], or [a = e, b = f, c = d], then Lemma 1 part (a) still holds, where S' is S with b decremented by 1 and e incremented by 1.*

Part (a) and its corollary tell us that it is generally not good to have strings of 2's of significantly different lengths. More specifically, whenever we have a string of two or more 2's we add the length of this string to the sum of the lengths of the adjacent strings of 2's. These numbers should be constant (or within 1) for all strings of two or more 2's in the schedule.

For example, if we have the schedule 12122222112121222222221121, we look at the two long strings of 2's and see that they have lengths 5 and 7. The adjacent strings of 2's have lengths 1 & 0 (for the string of length 5) and 2 & 1 (for the string of length 7), so our sums are 5+1+0=6 and 7+2+1=10. So, it is good to move a 2 from the string of length 7 to the one of length 5, giving the new schedule 12122222211212212222222121, with sums 6+1+0=7 and 6+2+1=9. By the Lemma, this new schedule is better. Since our sums differ by 2, we can do this again to get an even better schedule 12122222221121221222222121 with sums 8 and 8. However, if we repeat again, we get a schedule with sums 9 and 7, which is worse.

One thing the fact above allows us to do is eliminate adjacent long strings of 2's separated by a single 1. We do this by performing the operation in the lemma. We call our adjacent strings the b- and e-length strings. After multiple applications of the lemma (move a 2 from one of these strings to the other), we get one of the strings down to length 2. Essentially, we slide the 1 one way or the other to reduce the length of one string to 2 while increasing the length of the other.

As an example, consider the schedule 1212222212222112. We have adjacent strings of 2's of lengths 5 and 4. By considering these strings as we did above, we get sums of 5+1+4=10 and 4+5+0=9. So, if we move a 2 from the string of length 5 to the string of length 4, we improve our schedule. However, unlike before, our sums remain the same at 4+1+5=10 and 5+4+0=9. So,

we can keep doing this until we are left with only two 2's in the first string and a new schedule of 1212212222222112. We have basically slid the 1 to the left.

Part (b) is useful when we have blocks of 1's with some single 2's in them, bounded on both sides by at least two 2's. This lemma tells us that if the beginning or end of the block is 121, it is better to shift the 2 toward the inside of the block to get 112[*rest of block*] or [*rest of block*]211, provided there is a total of at least four 1's in the block.

An example of this is the schedule 12212111212112222. We have a block of 1's with only single 2's, which contains a total of seven 1's. This string is 1211121211, bordered on both sides by 22. The second part of the lemma tells us that it is better to have 112 or 211 than 121 on the beginning or end of this string. Since we start with 121, it is better to rearrange to get 1121121211, for a new and better schedule of 12211211212112222.

Part (c) tells us that large strings of 1's and 2's should never border each other. We get a better schedule by swapping the innermost 1 and 2 if we have at least three of each. A simple example of this is the schedule 121111122212. We would be better off using the schedule 121111212212, which swaps the inner 1 and 2 of the two long strings of 1's and 2's.

Part (d) tells us that reversing a schedule does not affect its expected waiting time. So, the schedules 121112122 and 221211121, for example, have the same expected waiting times.

Lemma 3 *(The Splitting Headache Lemma)*

Suppose schedule S is written as $s_1 s_2 \dots s_l$, where each s_i is either a 1 or a 2. Suppose also that there is a schedule $C = c_1 c_2 \dots c_k$, such that C has at least two 1's and at least two 2's, and $c_i = s_{m_1+i \bmod l} = s_{m_2+i \bmod l}$ $\forall i$, $1 \le i \le k$, for some $m_1 \ne m_2 \bmod l$. Then EWT(S, p_1) = $\frac{l_1}{l}$·EWT(S_1, p_1) + $\frac{l_2}{l}$·EWT(S_2, p_1), where $S_1 = s_{(m_1+1) \bmod l} s_{(m_1+2) \bmod l} \dots s_{(m_1+l_1) \bmod l}$, $S_2 = s_{(m_2+1) \bmod l} s_{(m_2+2) \bmod l} \dots s_{(m_2+l_2) \bmod l}$, $l_1 = (m_2 - m_1) \bmod l$, and $l_2 = (m_1 - m_2) \bmod l$.

This lemma tells us that under certain conditions we can split a schedule S into two schedules, S_1 and S_2. Schedule S will have an expected waiting time that is the weighted mean of the expected waiting times of S_1 and S_2. At any value of p_1, one of S_1 and S_2 will have a shorter expected waiting time and the other will have a longer expected waiting time. As a result, we should never use S, but instead choose the better of S_1 and S_2.

As an example, consider the schedule 11212122221122. We rewrite it (by shifting, since the schedule just cycles anyway) as 12121222211221. This is just 12 concatenated with 121222211221. Each of these starts with 1212, which is a string with two 1's and two 2's. It's easy to see this for the second schedule. For the first one, think of it not as 12, but as 121212..., which is what we broadcast when we use this schedule. This clearly starts with 1212. So, we can split our schedule into these two schedules. We can find the point p where the two schedules have the same expected waiting time, and see that 12 is better for $p_1 > p$ and 121222211221 is better for $p_1 < p$. So, instead of using the original schedule 11212122221122, we should use either 12 or 121222211221, depending on the value of p_1 relative to p.

3 Proof of Theorem 1

We begin the proof with an algorithm to improve schedules. This algorithm uses the Lemmas above to change schedules and reduce their expected waiting times. Most schedules can be reduced to better ones by the algorithm, but some cannot. We look only at the ones that cannot be reduced, since the rest are no better than one of these irreducible schedules. We then compare these irreducible schedules and find that a small subset of them (the schedules listed in Theorem 1) form the set of optimal schedules.

3.1 The Algorithm

The basic idea of the algorithm is to use the Rearrangement and Splitting Lemmas to modify arbitrary schedules and get better ones. We continue to modify schedules until we can no longer improve them with any sequence of actions described below. This gives us a smaller set of irreducible schedules. This set is just the set of schedules that cannot be improved by any sequence of actions described below. We say this set of schedules is fixed under these actions. Any action below will take a schedule in this set to another schedule in this set. We want to find this set.

Here is the algorithm to reduce arbitrary schedules to irreducible schedules:

Actions:

A1: $S = \mathbf{AB} \to S_1' = \mathbf{A}, S_2' = \mathbf{B}$, where both A and B start with the same subschedule C, which contains at least two 1's and two 2's.

A2: $S = \ldots 221\mathbf{21}\overbrace{(2)1(2)1\ldots(2)1}^{n\ 1's}22\ldots \to S' = \ldots 221\mathbf{12}\overbrace{(2)1(2)1\ldots(2)1}^{n\ 1's}22\ldots, n \geq 2$

A3: $S \to S' =$ reversal of S

A4: $S = [\ldots, a, \mathbf{n}, \mathbf{m}, b, \ldots] \to S' = [\ldots, a, \mathbf{2}, \mathbf{m{+}n{-}2}, b, \ldots], n \geq 2, m \geq 2, b \geq a$

A5: $S = [\ldots, a, \mathbf{b}, c, \ldots, d, \mathbf{e}, f, \ldots] \to S' = [\ldots, a, \mathbf{b{-}1}, c, \ldots, d, \mathbf{e{+}1}, f, \ldots], b \geq 3, e \geq 2$

A6: $S = \ldots 1\overbrace{22\ldots22}^{n\ 2's}\mathbf{2}\overbrace{11\ldots11}^{m\ 1's}2\ldots \to S' = \ldots 1\overbrace{22\ldots2}^{n-1}\mathbf{12}\overbrace{11\ldots1}^{m-1}2\ldots, n \geq 3, m \geq 3$

For any set of schedules S, do the following:

Step 1: Form the sets $V = S$ and $E = \{\}$.

Step 2: Repeat steps 3 and 4 until step 3 is no longer possible.

Step 3: Find an action from A1 through A6 that can be applied to some $v \in V$, resulting in a set of schedules W, such that $(v, w) \notin E$ for some $w \in W$.

Step 4: Let $V = V \cup W$ and $E = E \cup \{(v, w) | w \in W\}$.

The sets V and E that we end up with describe a directed graph, with an edge pointing from one node to another when the schedule corresponding to the first node can be modified (by some action) to give the schedule corresponding to the second node. From this graph, we take all nodes that are part of a strongly connected component of the graph. This is the set of irreducible schedules, since any other schedule can be reduced to one of them and none of them can be reduced to any other schedule that is strictly better.

3.2 The Irreducible Schedules

We use the notation "N" to mean "three or more". Different occurances of N within a schedule can correspond to different numbers greater than or equal to three. For example, [0,N,1,2,N] can represent the schedule [0,3,1,2,3], [0,3,1,2,4], or [0,10,1,2,12], but not [0,3,1,2,1], [0,0,1,2,2], or [0,10,1,2,2].

We will show a weaker version of Theorem 1.

Proposition 1 *All schedules can be reduced to one in the following list using actions A1 through A6 of the algorithm:*

$[0,1]$, $[0,1,1]$, $[0,1,2]$, $[0,1,2,N]$, $[0,1,N]$, $[0,1,N,2]$, $[0,1,N,2,N]$, $[0,2]$, $[0,2,1,N]$, $[0,2,N]$, $[0,2,N,1,N]$, $[0,N]$, $[0,N,1,2,N]$, $[0,N,1,N]$, $[0,N,1,N,2,N]$ $[0,N,2,N]$, $[1]$, $[1,2]$, $[1,2,N]$, $[1,N]$, $[1,N,2,N]$, $[2]$, $[2,N]$, $\overbrace{11\ldots1}^{m\geq3}221\overbrace{22\ldots2}^{n\geq3}112$, *and* $\overbrace{11\ldots1}^{m\geq3}221\overbrace{22\ldots2}^{n\geq3}11212$.

Note that the optimal schedules [0,2], [0,3], [0,4], [2], and [2,N] are all included in the list. We show later that these are the best of the schedules in the list.

For now, consider all schedules that don't contain both three consecutive 1's and three consecutive 2's. Without loss of generality, we assume there are no 111's. We will eliminate certain sets of schedules based on the fact that we can use one of the actions of the algorithm to find a better schedule. Since we can find a better schedule than any one in these classes, we know they are not in the fixed set of schedules under the operations of the algorithm.

First, we know that no irreducible schedule can contain more than one each of 1122, 2211, 1212, 2121, 1221, or 2112, since we could use action A1 on such a schedule to get a better schedule. Here, and in general, we ignore schedules that are optimal at exactly one point, such as the schedule before splitting, at the value of p_1 where the two better schedules have the same waiting time as the original. This is because the other two schedules are not only optimal at that point, but also in a neighborhood of that point. So, let's start eliminating bad schedules.

We can not have a schedule with two 0's in it, since we would have [... , 0, ... , 0, ...], which gives us two 2112's, or [... , 0, 0, ...], which gives us a 111. We can not have a schedule with two 2's in it, since we would have [... , 2, ... , 2, ...], which gives us two 1221's. If we have a schedule with two 1's in it, we have two instances of 121. If both are preceded by or followed by something other than 0, we get two instances of 1212 or 2121. The only way to prevent this is for one to be preceded by a 0 and the other to be followed by a 0. Since we can only have one 0, we need to have [... , 1, 0, 1, ...], with only 2's or larger in the rest of the schedule. But with 2's in the schedule, we can use action A2 to get a better schedule. So, the only schedule with two 1's is [0, 1, 1].

We can not have a schedule with two adjacent N's, where N represents some number greater than 2. If so, we could use action A4 to get a better schedule. So, we have [0,1,1] and all schedules that have at most one 0, 1, and 2, and no adjacent N's. It is straight-forward to list these:

[0,1,1], [N], [0], [0,N], [1], [1,N], [2], [2,N], [0,1], [0,1,N], [0,N,1], [0,N,1,N], [0,2], [0,2,N], [0,N,2], [0,N,2,N], [1,2], [1,2,N], [1,N,2], [1,N,2,N], [0,1,2], [0,1,2,N], [0,1,N,2], [0,N,1,2], [0,1,N,2,N], [0,N,1,2,N], [0,N,1,N,2], [0,2,1], [0,2,1,N], [0,2,N,1], [0,N,2,1], [0,2,N,1,N], [0,N,2,1,N], [0,N,2,N,1], [1,0,2], [1,0,2,N], [1,0,N,2], [1,N,0,2], [1,0,N,2,N], [1,N,0,2,N], [1,N,0,N,2], [1,2,0], [1,2,0,N], [1,2,N,0], [1,N,2,0], [1,2,N,0,N], [1,N,2,0,N], [1,N,2,N,0], [2,0,1], [2,0,1,N], [2,0,N,1], [2,N,0,1], [2,0,N,1,N], [2,N,0,1,N], [2,N,0,N,1], [2,1,0], [2,1,0,N], [2,1,N,0], [2,N,1,0], [2,1,N,0,N], [2,N,1,0,N], [2,N,1,N,0], [0,N,1,N,2,N], [0,N,2,N,1,N].

We can eliminate repetitions of the same schedule using periodicity of the schedules. For example, [0,2,N,1] = [1,0,2,N]. We can then use action A3 to eliminate reversals. We also eliminate [0], since it has no 2's. When we do this, we get the following list of schedules:

[0,1], [0,1,1], [0,1,2], [0,1,2,N], [0,1,N], [0,1,N,2], [0,1,N,2,N], [0,2], [0,2,1,N], [0,2,N], [0,2,N,1,N], [0,N], [0,N,1,2,N], [0,N,1,N], [0,N,1,N,2,N] [0,N,2,N], [1], [1,2], [1,2,N], [1,N], [1,N,2,N], [2], [2,N].

We omit a discussion of how to get the irreducible schedules when we have both a 111 and a 222 in the schedule. We will state without proof that the only such schedules that are not reducible are $\overbrace{11\ldots1}^{m\geq 3}221\overbrace{22\ldots2}^{n\geq 3}112$ and $\overbrace{11\ldots1}^{m\geq 3}221\overbrace{22\ldots2}^{n\geq 3}11212$.

So, these two schedules and the list above are the only possible schedules that can not be reduced by the actions of the algorithm, and Proposition 1 follows.

3.3 The Optimal Schedules

We have thus far shown that the schedules 1122, 11222, 112222, and $1221\overbrace{22\ldots22}^{n\geq 2}$ are irreducible. Now we finish the proof by showing they are better than the other irreducible schedules. We show

this by computation. We start with the schedules 1122, 11222, 112222, and all schedules of the form $1221\overbrace{22\ldots22}^{n\geq 2}$. Our goal is to show the following:

Proposition 2 *At any value of p_1, EWT(S, p_1) is minimized over all irreducible S by one of* 1122, 11222, 112222, *or* $1221\overbrace{22\ldots22}^{n\geq 2}$.

To show this, we first compute their expected waiting times as a function of p_1. We then plot expected waiting time vs. p_1 for each schedule. We find the appropriate intersection points and see that 1122 is best on $p_1 \in \left(\frac{5}{16}, \frac{1}{2}\right)$, 11222 is best on $\left(\frac{5}{21}, \frac{5}{16}\right)$, 112222 is best on $\left(\frac{1}{5}, \frac{5}{21}\right)$, 122122 is best on $\left(\frac{2}{17}, \frac{1}{5}\right)$, and $12212\overbrace{22\ldots22}^{n\geq 1}$ is best on $\left(\frac{8}{(n+1)^2+13(n+1)+68}, \frac{8}{n^2+13n+68}\right)$. This is illustrated in Figure 2.

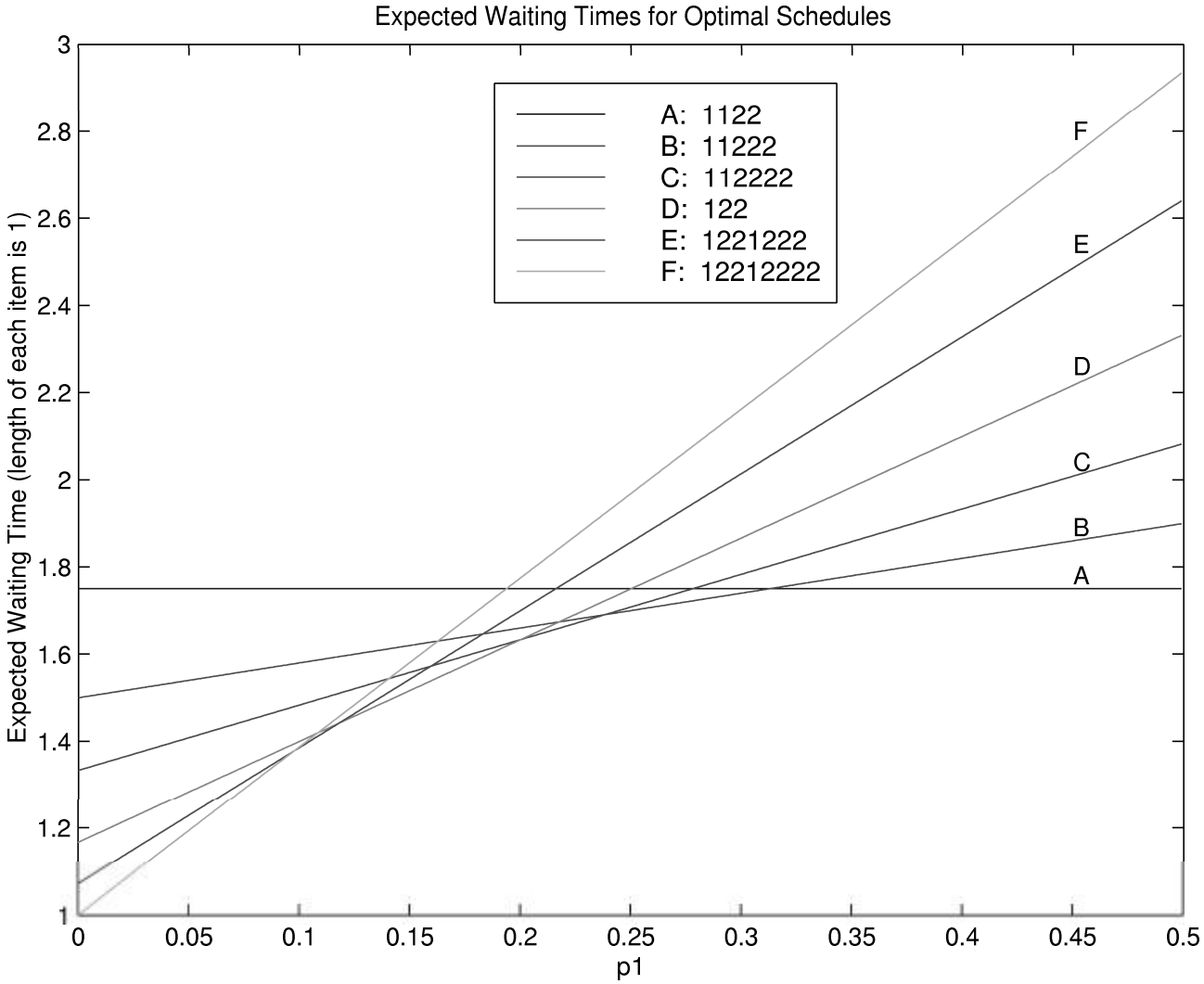

Figure 2: Expected Waiting time vs. p_1 for some of the optimal schedules

It remains to show that any other irreducible schedule is worse than one of these schedules for any value of p_1. Suppose there is some schedule, with graph defined by $t = a \cdot p_1 + b$, that has a shorter expected waiting time at some p_1 than any of our optimal schedules. Then it will dip below the (piecewise linear) MIN function of the graphs. Since this function is linear, it must be less at some corner of the MIN function. So, if we can show that no schedule has a graph that dips below a corner, we have shown that our schedules are optimal.

We do this by checking the set of intersection points of 1122 & 11222, 11222 & 112222, and 112222 & 122, which is $C_1 = \left\{\left(\frac{5}{16}, \frac{7}{4}\right), \left(\frac{5}{21}, \frac{71}{42}\right), \left(\frac{1}{5}, \frac{49}{30}\right)\right\}$, and the set of points $C_2 = \left\{\left(\frac{8}{n^2+13n+68}, \frac{1}{2} + \frac{8(n^2+14n+48)}{n^3+19n^2+146n+408}\right) \middle| n \geq 0\right\}$, where $122122\overbrace{2\ldots2}^{n}$ and $122122\overbrace{2\ldots2}^{n+1}$ intersect, $\forall n \geq 0$. These are our corner points. We evaluate $t = a \cdot p_1 + b$ at the corner point (p_c, t_c) and compute the difference $t - t_c = a \cdot p_c + b - t_c$. We want to show this is always positive. We do this by expressing the waiting times in terms of parameters and working with the resulting expressions to show they are positive for all values of the parameters.

As an example, consider the schedule [0,N]. If $m = N - 2$, this has expected waiting time $\frac{m^2+7m+10}{4(m+4)}$ for item 1 and $\frac{5}{4(m+4)}$ for item 2. So, $t - t_c$ at the "n^{th}" corner point in C_2 is $\frac{m^2+7m}{4m+16} \cdot \frac{8}{n^2+13n+68} + \frac{5}{2m+8} - \frac{4(n^2+14n+48)}{n^3+19n^2+146n+408}$. This is non-negative $\Leftrightarrow 4(m^2+7m)(n+6) + 5(n^3+19n^2+146n+408) - 4(2m+8)(n^2+14n+48) \geq 0$. We rewrite as a quadratic in m to get $(4n+24)m^2 - (8n^2+84n+216)m + (5n^3+63n^2+282n+504) \geq 0$. We know this is always positive if it has no real roots, so it is always positive if its discriminant is negative. So, we want $4(4n+24)(5n^3+63n^2+282n+504) - (8n^2+84n+216)^2 > 0$. Simplifying, we get $16n^4+144n^3+48n^2-1152n+1728 > 0$. This is true for $n = 0, 1, 2$. For $n \geq 3$, note that this expression is greater than $16(n-3)^4$, which is non-negative for all $n \geq 3$. So, this expression is always positive, and hence there are no values of $m \geq 0$ and $n \geq 0$ where the schedule [0,N] is better than $1221222\overbrace{\ldots2}^{n}$. We also check against 1122, 11222, 112222, and 122122, and see that we do not dip below their corner points. We do this the same way, except now we only have the single parameter n instead of both m and n.

We use the same basic idea for all other schedules without both 111's and 222's. Some of them have more than one group of N 2's. For these, we use action A5 to determine how many 2's the blocks can have relative to each other. It is generally within a small number of the others, and we consider all possible combinations, doing the above calculation for each. For the schedules with both 111 and 222 we use a different method. We omit its discussion and simply state that none of these schedules is better than our list of optimal schedules for any value of p_1. So, Proposition 2 is proved, and combining with Proposition 1 we see that our small set of schedules $\{1122, 11222, 112222, 1221\overbrace{22\ldots22}^{n\geq 2}\}$ performs better than any other schedule. The intervals on which each is optimal are described by C_1 and C_2. These agree with those in the statement of Theorem 1, thus completing the proof.

4 Future Work

Future work involves finding the best schedules for more than two items, and for items of varying lengths. We have shown that a single split can improve expected waiting time. More work can be done to see the effects of multiple splits on expected waiting time. We would also like to look at the effects of transmission errors and ways to effectively combat them, as well as adding multiple transmission channels to improve bandwidth.

References

[1] D. Aksoy, M. J. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting", IEEE INFOCOM, San Francisco, CA, March 1998.

[2] A. Bar-Noy, R. Bhatia, J. Naor, B. Schieber, "Minimizing Service and Operations Cost of Periodic Scheduling", 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1998.

[3] A. Bestavros, "AIDA-based Real-Time Fault-Tolerant Broadcast Disks", Technical Report 96-001, Computer Science, Boston University, January 5, 1996.

[4] M. J. Franklin, S. Zdonik, "Dissemination-Based Information Systems", IEEE Data Engineering Bulletin, Vol 19, No 3, September 1996.

[5] S. Hameed, *Scheduling Information Broadcast in Asymmetric Environment*, Masters of Science thesis, Texas A&M University, May 1997.

[6] S. Hameed, N. H. Vaidya, "Log-time Algorithms for Scheduling Single and Multiple Channel Data Broadcast", MOBICOM'97, Budapest, September 1997.

[7] S. Jiang, N. H. Vaidya, "Scheduling Algorithms for a Data Broadcast System: Minimizing Variance of the Response Time", Technical Report 98-005, Computer Science, Texas A&M University, February 4, 1998.

[8] H. V. Leong, A. Si, "Database Caching Over the Air-Storage", *The Computer Journal* 40(7): 401-415, 1997.

[9] N. H. Vaidya, S. Hameed, "Data Broadcast in Asymmetric Wireless Environments", First International Workshop on Satellite-based Information Services (WOSBIS), Rye, NY, November 1996.

[10] S. Zdonik, M. J. Franklin, R. Alonso, S. Acharya, "Are 'Disks in the Air' Just 'Pie in the Sky'?", IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994.

A Appendix: Proof of the Lemmas

Proof of Lemma 1

The idea is to compute the expected waiting times for each schedule and compare them to see which is better. To compute the expected waiting times, we start by computing the waiting times starting at each block. We then find the average waiting time for the entire schedule by dividing the sum of these waiting times by the schedule's length. Then we add $\frac{1}{2}$, for the expected time we wait when we initially start listening for the beginning of the first full block. We do this separately for items 1 and 2, and weight these times by p_1 and p_2 to get the overall expected time.

part (a)

The expected number of blocks to wait through to get two blocks of item 1 using schedule S is

$$t_1 = \frac{1}{2} + \frac{\cdots + \sum_{i=0}^{a}(b+i) + \sum_{i=0}^{b}(c+i) + \cdots + \sum_{i=0}^{d}(e+i) + \sum_{i=0}^{e}(f+i) + \cdots}{length\ of\ schedule}$$

The corresponding value for S' is

$$t'_1 = \frac{1}{2} + \frac{\cdots + \sum_{i=0}^{a}(b-1+i) + \sum_{i=0}^{b-1}(c+i) + \cdots + \sum_{i=0}^{d}(e+1+i) + \sum_{i=0}^{e+1}(f+i) + \cdots}{length\ of\ schedule}$$

The difference between these two times is

$$\begin{aligned} \Delta t_1 &= t'_1 - t_1 \\ &= \frac{(-1)\cdot(a+1) - (b+c) + (1)\cdot(d+1) + (e+f+1)}{length\ of\ schedule} \\ &= \frac{(d+e+f) - (a+b+c) + 1}{length\ of\ schedule} \end{aligned}$$

The expected waiting time for item 2 is the same for both schedules, so $\Delta t_2 = 0$.

The overall change in waiting time is $p_1 \cdot \Delta t_1 + p_2 \cdot \Delta t_2 = p_1 \cdot \frac{(d+e+f)-(a+b+c)+1}{length\ of\ schedule}$. Since p_1 and *length of schedule* are both positive, if $(a+b+c)-(d+e+f) \geq 1$ then $\Delta t_1 \leq 0$, and the expected waiting time for S' is less than or equal to the expected waiting time for S.

part (b)

The expected number of blocks to wait through to get two blocks of item 1 using schedule T is

$$t_1 = \frac{1}{2} + \frac{\cdots + \sum_{i=1}^{n+1} i + (1+x) + (x) + (x+y) + \cdots}{length\ of\ schedule}$$

Using schedule T' we get

$$t'_1 = \frac{1}{2} + \frac{\cdots + \sum_{i=0}^{n} i + (1+x) + (1+x+y) + (x+y) + \cdots}{length\ of\ schedule}$$

The expected number of blocks to wait through to get two blocks of item 2 using schedule T is

$$t_2 = \frac{1}{2} + \frac{\cdots + (1) + (2+Q) + (1+Q) + (1+P) + (P) + \cdots}{length\ of\ schedule}$$

Using schedule T' we get

$$t'_2 = \frac{1}{2} + \frac{\cdots + (2) + (2+Q) + (1+Q) + (Q) + (P) + \cdots}{length\ of\ schedule}$$

The differences are:

$$\Delta t_1 = t'_1 - t_1 = \frac{-(n+1) + (1+y)}{length\ of\ schedule} = \frac{y-n}{length\ of\ schedule} < 0$$

$$\Delta t_2 = t'_2 - t_2 = \frac{1 + (Q) - (P+1)}{length\ of\ schedule} = \frac{Q-P}{length\ of\ schedule} \leq 0$$

Here, Q is the number of 1's before the next 2, starting from the position just before the x 2's, and P is the number of 1's before the second 2, starting from the same position. Since Δt_1 and Δt_2 are both non-positive, so is Δt, and the expected waiting time for schedule T' is less than or equal to the expected waiting time for schedule T.

part (c)

The expected number of blocks to wait through to get two blocks of item 1 using schedule U is

$$t_1 = \frac{1}{2} + \frac{\cdots + (t-2)\cdot 0 + u + \sum_{i=0}^{u}(P+i) + \cdots}{length\ of\ schedule}$$

Using schedule U' we get

$$t'_1 = \frac{1}{2} + \frac{\cdots + (t-3)\cdot 0 + 1 + u + (u-1) + \sum_{i=0}^{u-1}(P+i) + \cdots}{length\ of\ schedule}$$

where P is the number of blocks to wait through to get two blocks of item 1, starting after the last of the u 2's.

The expected number of blocks to wait through to get two blocks of item 2 using schedule U is

$$t_2 = \frac{1}{2} + \frac{\cdots + \sum_{i=0}^{s}(i\,[+t]_{s=1}) + [(s-2)\cdot 0 + t]_{s>1} + t + \sum_{i=0}^{t} i + \cdots}{length\ of\ schedule}$$

Using schedule U' we get

$$t'_2 = \frac{1}{2} + \frac{\cdots + \sum_{i=0}^{r}(i-1\,[+t]_{s=1})\,[+(s-2)\cdot 0 + (t-1)]_{s>1} + \sum_{i=0}^{t} i + 1 + \cdots}{length\ of\ schedule}$$

Here, bracketed expressions with subscripts are evaluated as follows: if the subscript expression is true, evaluate the expression inside; if it is false, remove the entire bracketed expression from the equation.

The differences are:

$$\Delta t_1 = t'_1 - t_1 = \frac{(1-0) + ((u-1) - (P+u))}{length\ of\ schedule} = \frac{-P}{length\ of\ schedule} \leq 0$$

$$\Delta t_2 = t'_2 - t_2 = \frac{-1\,[\cdot (r+1)]_{s=1} + 1}{length\ of\ schedule} = \frac{0\,[-r]_{s=1}}{length\ of\ schedule} \leq 0$$

Since Δt_1 and Δt_2 are both non-positive, so is Δt, so the expected waiting time for schedule U' is less than or equal to the expected waiting time for schedule U.

part (d)

Any schedule is just a bunch of 1's and 2's, where each consecutive pair of 1's is separated by some non-negative number of 2's. So our schedule V is just:

$$V = 1\overbrace{2\ldots2}^{n_1}1\overbrace{2\ldots2}^{n_2}1\ldots1\overbrace{2\ldots2}^{n_{m-1}}1\overbrace{2\ldots2}^{n_m}$$

The expected waiting time for item 1 is:

$$\begin{aligned}
t_1 &= \sum_{i=0}^{n_1}(n_2+i)+\sum_{i=0}^{n_2}(n_3+i)+\cdots+\sum_{i=0}^{n_{m-1}}(n_m+i)+\sum_{i=0}^{n_m}(n_1+i) \\
&= \sum_{i=0}^{m}(n_i+1)\,n_{i+1}+\sum_{i=0}^{m}\frac{n_i\,(n_i+1)}{2} \\
&= \sum_{i=0}^{m}n_in_{i+1}+\sum_{i=0}^{m}n_i+\sum_{i=0}^{m}\frac{n_i\,(n_i+1)}{2} \\
&= \sum_{i=0}^{m}n_{i-1}n_i+\sum_{i=0}^{m}n_{i-1}+\sum_{i=0}^{m}\frac{n_i\,(n_i+1)}{2} \\
&= \sum_{i=0}^{m}(n_i+1)\,n_{i-1}+\sum_{i=0}^{m}\frac{n_i\,(n_i+1)}{2} \\
&= \sum_{i=0}^{n_m}(n_{m-1}+i)+\sum_{i=0}^{n_{m-1}}(n_{m-2}+i)+\cdots+\sum_{i=0}^{n_2}(n_1+i)+\sum_{i=0}^{n_1}(n_m+i)
\end{aligned}$$

This is just the formula for the expected waiting time for item 1 using V'. So V and V' have the same expected waiting times for item 1. Similarly they have the same waiting times for item 2. So, the expected waiting times for V and V' are equal.

Proof of Lemma 2

We use the same proof. However, with overlap, we must equate different variables, depending on which sections overlap. The change in the proof is to substitute the new values of equivalent variables in place of their old values. For example, if $b = d$ and $c = e$, we must now use $c + 1$ instead of c and $d - 1$ instead of d, since $c = e$ and $b = d$ and we now have $e + 1$ and $b - 1$ instead of e and b. When we work out the math for the new times, we see that the proof still works.

Proof of Lemma 3

We can write schedule S as

$$(*)\ \overbrace{\overbrace{|_{C_{1(start)}}\cdots\cdots}^{S_1}\overbrace{|_{C_{2(start)}}\cdots\cdots}^{S_2}}^{S}\overbrace{|_{C_{1(start)}}\cdots\cdots}^{S_1}$$

where $C_{1(start)}$ indicates the start of the first instance of subschedule C within S and $C_{2(start)}$ indicates the start of the second occurance. Since S repeats cyclically, we can assume without loss of generality that it starts with one of the subschedules C.

We can also write schedules S_1 and S_2 as follows:

$$(**)\ \overbrace{|_{C_{(start)}}\cdots\cdots}^{S_1}\overbrace{|_{C_{(start)}}\cdots\cdots}^{S_1}$$

$$(***)\ \overbrace{|_{C_{(start)}}\cdots\cdots}^{S_2}\overbrace{|_{C_{(start)}}\cdots\cdots}^{S_2}$$

If we start waiting at some time within the first S_1 group in either (*) or (**), we will wait through a certain number of blocks before finding enough blocks of the desired item. If we stay within the initial S_1 group, the times are identical for (*) and (**) since the sequence of blocks that we encounter is the same in each case. If we must proceed into the next group, we first enter the C subschedule in either case. Since C contains two blocks each of items 1 and 2, we will not have to advance past the C group, so the waiting times in these cases are identical, since we again encounter the same sequence of blocks. So, it follows that the waiting time for schedule S, given that we arrive somewhere within subschedule S_1, is the same as it is for S_1 treated as a full schedule. The same applies for subschedule S_2. So, we have $EWT(S, p_1) = Pr(arrive\ in\ S_1) \cdot EWT(S_1, p_1) + Pr(arrive\ in\ S_2) \cdot EWT(S_2, p_1)$. The probabilities of arriving in subschedules S_1 and S_2 within schedule S are simply $\frac{l_1}{l}$ and $\frac{l_2}{l}$, respectively. So, we have $EWT(S, p_1) = \frac{l_1}{l} \cdot EWT(S_1, p_1) + \frac{l_2}{l} \cdot EWT(S_2, p_1)$.